



L7: Memory Basics and Timing



Acknowledgement: Nathan Ickes



Memory Classification & Metrics



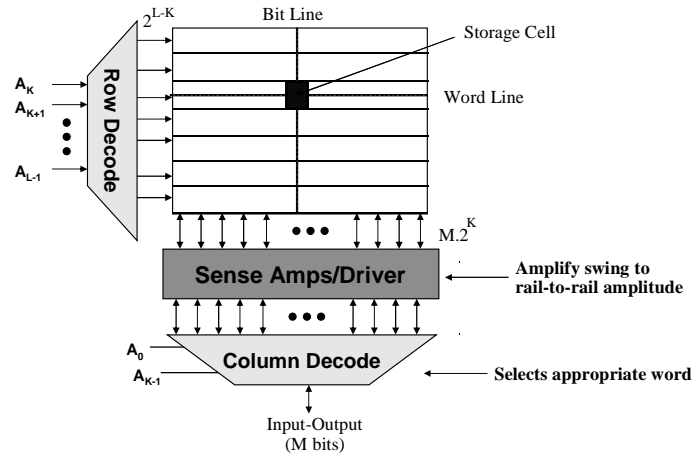
Read-Write Memory		Non-Volatile Read-Write Memory	Read-Only Memory
Random Access	Non-Random Access	EPROM E ² PROM	Mask-Programmed
SRAM DRAM	FIFO LIFO	FLASH	

Key Design Metrics:

1. Memory Density (number of bits/ μm^2) and Size
2. Access Time (time to read or write) and Throughput
3. Power Dissipation



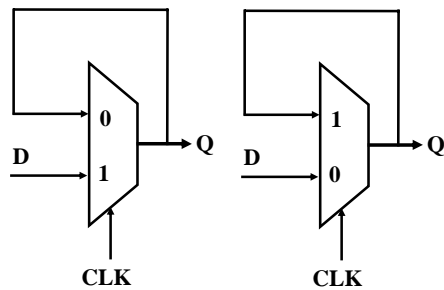
Memory Array Architecture



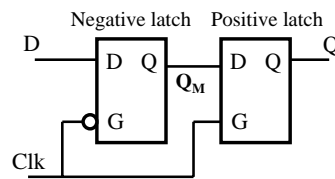
Latch and Register Based Memory



Positive Latch Negative Latch



Register Memory

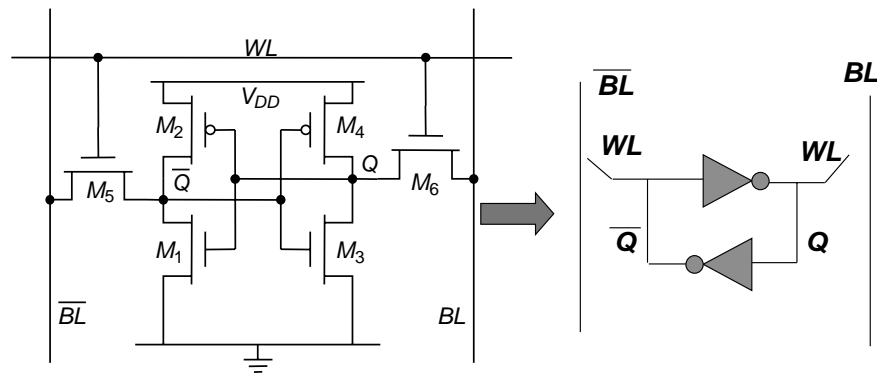


- Works fine for small memory blocks (e.g., small register files)
- Inefficient in area for large memories – density is the key metric in large memory circuits

How do we minimize cell size?



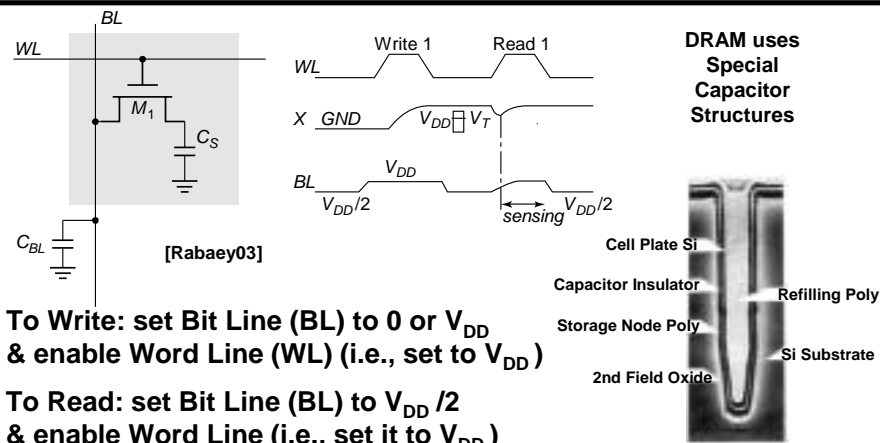
Static RAM (SRAM) Cell (The 6-T Cell)



- State held by cross-coupled inverters (M1-M4)
- Retains state as long as power supply turned on
- Feedback must be overdriven to write into the memory



Dynamic RAM (DRAM) Cell



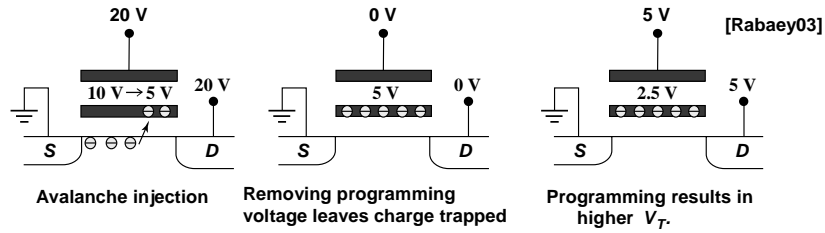
DRAM uses Special Capacitor Structures

To Write: set Bit Line (BL) to 0 or V_{DD} & enable Word Line (WL) (i.e., set to V_{DD})

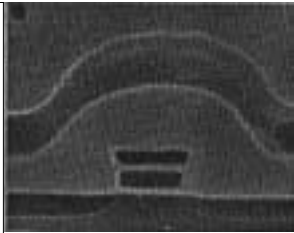
To Read: set Bit Line (BL) to $V_{DD}/2$ & enable Word Line (i.e., set it to V_{DD})

- DRAM relies on charge stored in a capacitor to hold state
- Found in all high density memories (one bit/transistor)
- Must be “refreshed” or state will be lost – high overhead

EPROM Cell – The Floating Gate Transistor

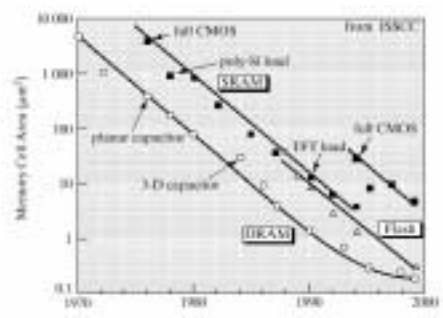
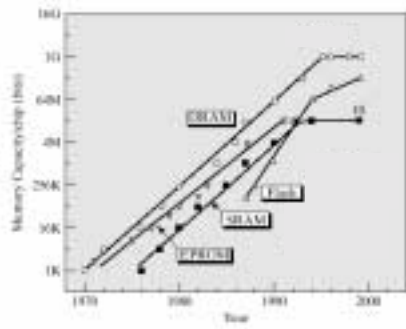


EPROM Cell
Courtesy Intel



This is a non-volatile memory (retains state when supply turned off)

Density/Size Memory Trends



From [Itoh01]



Key Messages on Memory Devices



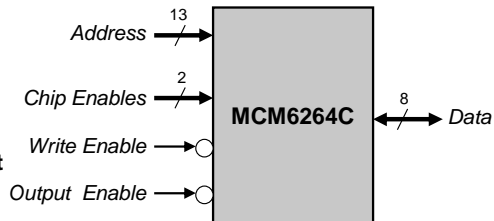
- **SRAM vs. DRAM**
 - SRAM holds state as long as power supply is turned on. DRAM must be “refreshed” – results in more complicated control
 - DRAM has much higher density, but requires special capacitor technology.
 - FPGA usually implemented in a standard digital process technology and uses SRAM technology
- **Non-Volatile Memory**
 - Fast Read, but very slow write (EPROM must be removed from the system for programming!)
 - Holds state even if the power supply is turned off
- **Memory Internals**
 - Has quite a bit of analog circuits internally -- pay particular attention to noise and PCB board integration
- **Device details**
 - Don't worry about them, wait until 6.012, 6.371 or 6.374



MCM6264C 8k x 8 Static RAM



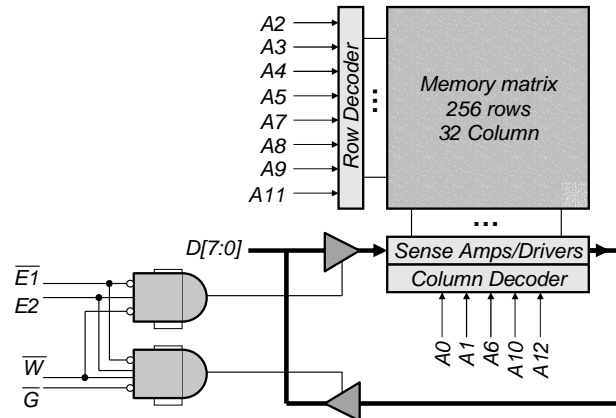
- **Same (bidirectional) data bus used for reading and writing**
- **Chip Enables ($\overline{E1}$ and $E2$)**
 - $\overline{E1}$ must be low and $E2$ must be high to enable the chip
- **Write Enable (\overline{W})**
 - When low (and chip is enabled), the values on the data bus are written to the location selected by the address bus
- **Output Enable (\overline{G})**
 - When low (and chip is enabled), the data bus is driven with the value of the selected memory location



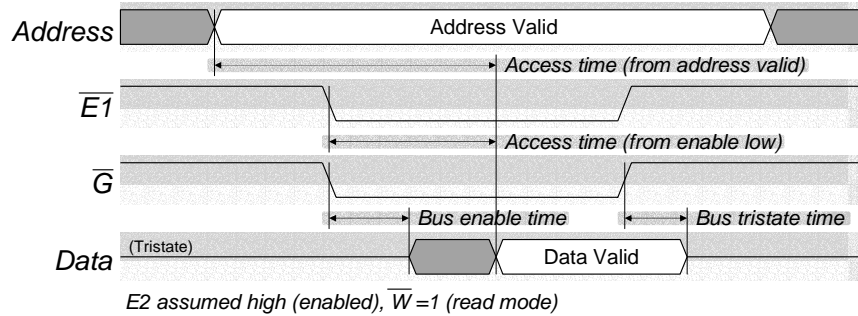
HE0	0	00	VCC
A0	1	01	W
A0	2	01	W
A0	3	01	W
A0	4	01	W
A0	5	01	W
A0	6	01	W
A0	7	01	W
A0	8	01	W
A0	9	01	W
A0	10	01	W
A0	11	01	W
A0	12	01	W
A0	13	01	W
A0	14	01	W
A0	15	01	W
A0	16	01	W
A0	17	01	W
A0	18	01	W
A0	19	01	W
A0	20	01	W
A0	21	01	W
A0	22	01	W
A0	23	01	W
A0	24	01	W
A0	25	01	W
A0	26	01	W
A0	27	01	W
A0	28	01	W
A0	29	01	W
A0	30	01	W
A0	31	01	W
A0	32	01	W
A0	33	01	W
A0	34	01	W
A0	35	01	W
A0	36	01	W
A0	37	01	W
A0	38	01	W
A0	39	01	W
A0	40	01	W
A0	41	01	W
A0	42	01	W
A0	43	01	W
A0	44	01	W
A0	45	01	W
A0	46	01	W
A0	47	01	W
A0	48	01	W
A0	49	01	W
A0	50	01	W
A0	51	01	W
A0	52	01	W
A0	53	01	W
A0	54	01	W
A0	55	01	W
A0	56	01	W
A0	57	01	W
A0	58	01	W
A0	59	01	W
A0	60	01	W
A0	61	01	W
A0	62	01	W
A0	63	01	W
A0	64	01	W
A0	65	01	W
A0	66	01	W
A0	67	01	W
A0	68	01	W
A0	69	01	W
A0	70	01	W
A0	71	01	W
A0	72	01	W
A0	73	01	W
A0	74	01	W
A0	75	01	W
A0	76	01	W
A0	77	01	W
A0	78	01	W
A0	79	01	W
A0	80	01	W
A0	81	01	W
A0	82	01	W
A0	83	01	W
A0	84	01	W
A0	85	01	W
A0	86	01	W
A0	87	01	W
A0	88	01	W
A0	89	01	W
A0	90	01	W
A0	91	01	W
A0	92	01	W
A0	93	01	W
A0	94	01	W
A0	95	01	W
A0	96	01	W
A0	97	01	W
A0	98	01	W
A0	99	01	W
A0	100	01	W
A0	101	01	W
A0	102	01	W
A0	103	01	W
A0	104	01	W
A0	105	01	W
A0	106	01	W
A0	107	01	W
A0	108	01	W
A0	109	01	W
A0	110	01	W
A0	111	01	W
A0	112	01	W
A0	113	01	W
A0	114	01	W
A0	115	01	W
A0	116	01	W
A0	117	01	W
A0	118	01	W
A0	119	01	W
A0	120	01	W
A0	121	01	W
A0	122	01	W
A0	123	01	W
A0	124	01	W
A0	125	01	W
A0	126	01	W
A0	127	01	W
A0	128	01	W
A0	129	01	W
A0	130	01	W
A0	131	01	W
A0	132	01	W
A0	133	01	W
A0	134	01	W
A0	135	01	W
A0	136	01	W
A0	137	01	W
A0	138	01	W
A0	139	01	W
A0	140	01	W
A0	141	01	W
A0	142	01	W
A0	143	01	W
A0	144	01	W
A0	145	01	W
A0	146	01	W
A0	147	01	W
A0	148	01	W
A0	149	01	W
A0	150	01	W
A0	151	01	W
A0	152	01	W
A0	153	01	W
A0	154	01	W
A0	155	01	W
A0	156	01	W
A0	157	01	W
A0	158	01	W
A0	159	01	W
A0	160	01	W
A0	161	01	W
A0	162	01	W
A0	163	01	W
A0	164	01	W
A0	165	01	W
A0	166	01	W
A0	167	01	W
A0	168	01	W
A0	169	01	W
A0	170	01	W
A0	171	01	W
A0	172	01	W
A0	173	01	W
A0	174	01	W
A0	175	01	W
A0	176	01	W
A0	177	01	W
A0	178	01	W
A0	179	01	W
A0	180	01	W
A0	181	01	W
A0	182	01	W
A0	183	01	W
A0	184	01	W
A0	185	01	W
A0	186	01	W
A0	187	01	W
A0	188	01	W
A0	189	01	W
A0	190	01	W
A0	191	01	W
A0	192	01	W
A0	193	01	W
A0	194	01	W
A0	195	01	W
A0	196	01	W
A0	197	01	W
A0	198	01	W
A0	199	01	W
A0	200	01	W
A0	201	01	W
A0	202	01	W
A0	203	01	W
A0	204	01	W
A0	205	01	W
A0	206	01	W
A0	207	01	W
A0	208	01	W
A0	209	01	W
A0	210	01	W
A0	211	01	W
A0	212	01	W
A0	213	01	W
A0	214	01	W
A0	215	01	W
A0	216	01	W
A0	217	01	W
A0	218	01	W
A0	219	01	W
A0	220	01	W
A0	221	01	W
A0	222	01	W
A0	223	01	W
A0	224	01	W
A0	225	01	W
A0	226	01	W
A0	227	01	W
A0	228	01	W
A0	229	01	W
A0	230	01	W
A0	231	01	W
A0	232	01	W
A0	233	01	W
A0	234	01	W
A0	235	01	W
A0	236	01	W
A0	237	01	W
A0	238	01	W
A0	239	01	W
A0	240	01	W
A0	241	01	W
A0	242	01	W
A0	243	01	W
A0	244	01	W
A0	245	01	W
A0	246	01	W
A0	247	01	W
A0	248	01	W
A0	249	01	W
A0	250	01	W
A0	251	01	W
A0	252	01	W
A0	253	01	W
A0	254	01	W
A0	255	01	W



Inside the '6264C



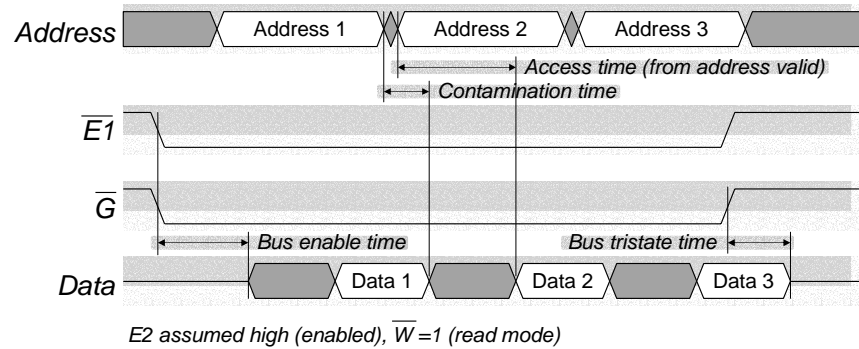
Reading From SRAM



- Read cycle begins when all enable signals ($\overline{E1}$, $E2$, \overline{G}) are active
- Data is valid after read access time
 - Access time is indicated by full part number: MCM6264CP-12 \rightarrow 12ns
- Data bus is tristated shortly after \overline{G} or $\overline{E1}$ goes high



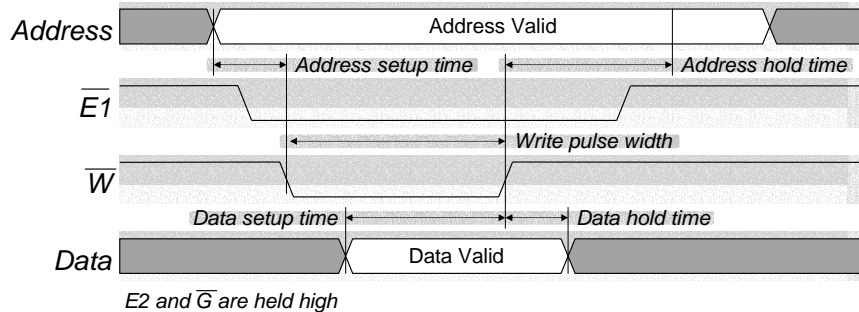
Address Controlled Reads



- Can perform multiple reads without disabling chip
- Data bus follows address bus, after some delay



Writing to SRAM



- Data latched when \bar{W} or $\bar{E1}$ goes high (or E2 goes low)
 - Data must be stable at this time
 - Address must be stable before \bar{W} goes low
- Write waveforms are more important than read waveforms
 - Glitches can cause writes to random addresses!

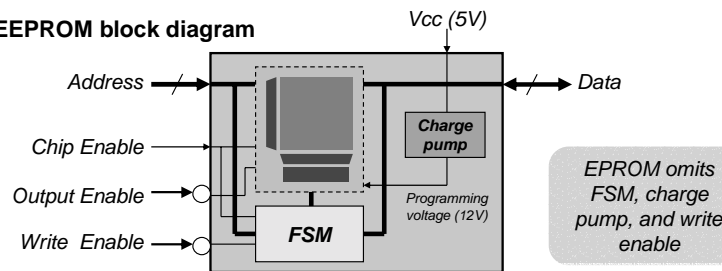


Flash and (E)EPROM

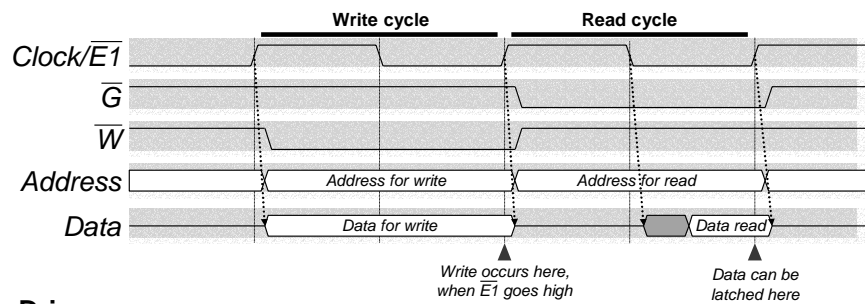


- Reading from flash or (E)EPROM is the same as reading from SRAM
- Vpp: input for programming voltage (12V)
 - EPROM: Vpp is supplied by programming machine
 - Modern flash/EEPROM devices generate 12V using an on-chip charge pump
- EPROM lacks a write enable
 - Not in-system programmable (must use a special programming machine)
- For flash and EEPROM, write sequence is controlled by an internal FSM
 - Writes to device are used to send signals to the FSM
 - Although the same signals are used, one can't write to flash/EEPROM in the same manner as SRAM

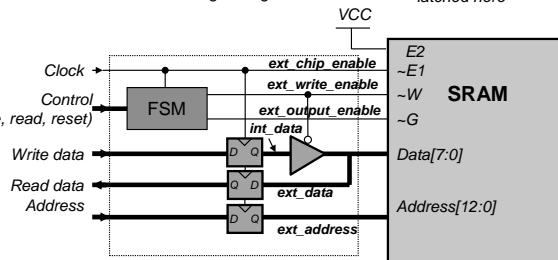
Flash/EEPROM block diagram



Sample Memory Interface Logic



- Drive memory enable with clock
 - Ensures data and memory busses are stable for writes
 - Minimum clock period is twice memory access time





Toy Memory Interface in VHDL (I)



```
library ieee;
use ieee.std_logic_1164.all;

entity mem_int is
  port (clock : in std_logic;
        reset : in std_logic;
        write : in std_logic;
        read : in std_logic;
        address : in std_logic_vector(12 downto 0);
        write_data : in std_logic_vector(7 downto 0);
        read_data : out std_logic_vector(7 downto 0);
        ext_chip_enable : out std_logic;
        ext_write_enable : out std_logic;
        ext_output_enable : out std_logic;
        ext_address : out std_logic_vector(12 downto 0);
        ext_data : inout std_logic_vector(7 downto 0));
end mem_int;

architecture behavioral of mem_int is

  signal int_data : std_logic_vector(7 downto 0);

  (cont. on next viewgraph)
```



Toy Memory Interface in VHDL (II)



```
begin
  ext_chip_enable <= clock;
  ext_data <= int_data when ext_write_enable = '0' else (others => 'Z');

  process (clock, reset)
  begin
    if reset = '0' then
      ext_write_enable <= '1';
      ext_output_enable <= '1';
    elseif clock'event and clock = '1' then
      ext_address <= address;
      read_data <= ext_data;
      int_data <= write_data;
      if write = '1' then
        ext_write_enable <= '0';
      elseif read = '1' then
        ext_output_enable <= '0';
      else
        ext_write_enable <= '1';
        ext_output_enable <= '1';
      end if;
    end if;
  end process;

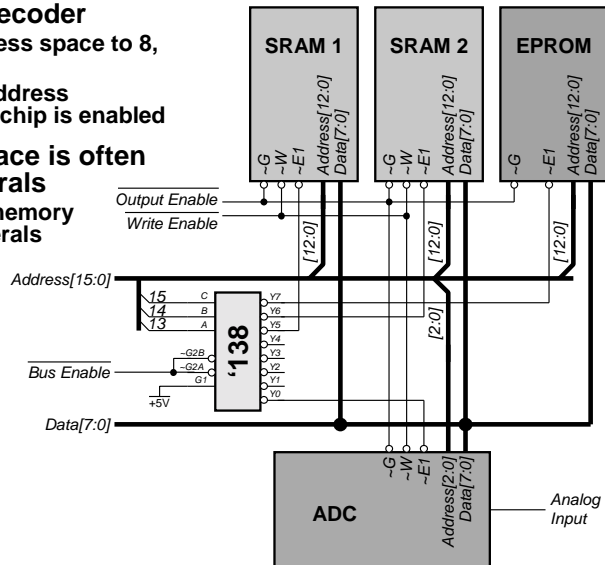
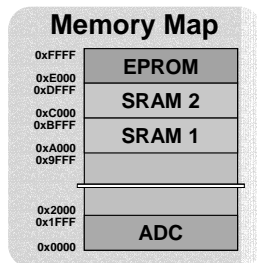
end behavioral;
```



Creating Larger Memories



- '138 is a 3-to-8 decoder
 - Maps 16 bit address space to 8, 13 bit segments
 - Upper 3 bits of address determine which chip is enabled
- SRAM-like interface is often used for peripherals
 - Referred to as "memory mapped" peripherals



L7: 6.111 Spring 2003

Introductory Digital Systems Laboratory

19



Testing Memories



- **Common device problems**
 - Bad locations: rare for individual locations to be bad
 - Slow (out-of-spec) timings: causes intermittent failures
 - Catastrophic device failure: e.g., ESD
 - Missing wire-bonds/devices (!): possible with automated assembly
 - Transient Failures: Alpha particles, power supply glitch
- **Common circuit problems**
 - Stuck-at-Faults: a pin shorted to VDD or GND
 - Open Circuit Fault: connections unintentionally left out
 - Open or shorted address wires: causes data to be written to incorrect locations
 - Open or shorted control wires: generally renders memory completely inoperable

L7: 6.111 Spring 2003

Introductory Digital Systems Laboratory

20



Testing Memory



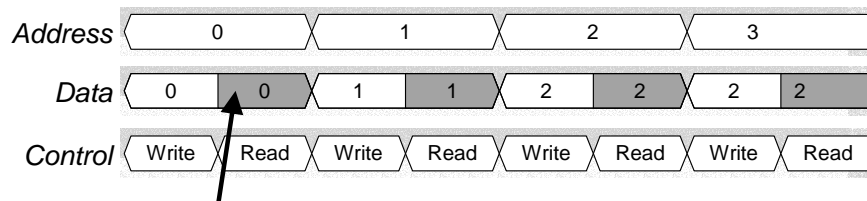
- Since device problems generally affect the entire chip, almost any test will detect them
- Writing (and reading back) many different data patterns can detect data bus problems
- Writing unique data to every location and then reading it back can detect address bus problems



Testing Memory



- An idea that almost works
 1. Write 0 to location 0
 2. Read location 0, compare value read with 0
 3. Write 1 to location 1
 4. Read location 1, compare value read with 1
 5. ...
- What is the problem?
 - Suppose the memory was missing (or output enable was disconnected)



Data bus is undriven but wire capacitance briefly maintains the bus state: memory appears to be ok!



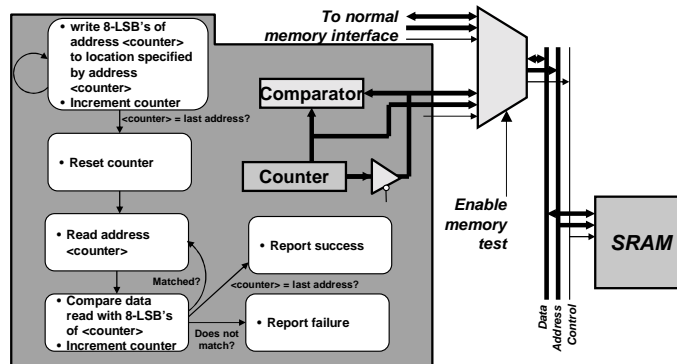
Testing Memory



Write to all locations, then read back all locations

- Separates read/write to the same location with reads/writes of different data to different locations
- (both data and address busses are changed between read and write to same location)

- Write 0 to address 0
- Write 1 to address 1
- ...
- Write ($n \bmod 256$) to address n
- Read address 0, compare with 0
- Read address 1, compare with 1
- ...
- Read address n , compare with ($n \bmod 256$)



Multi-Port Memories

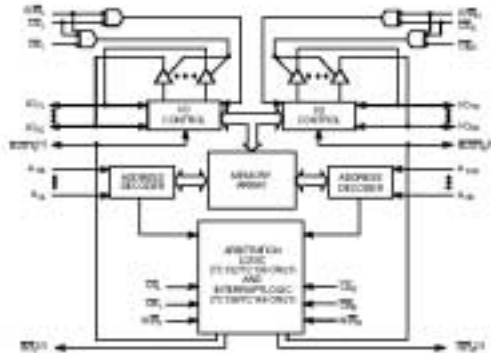


Applications

- Register files, FIFO, buffer, etc.

Cypress CY7C132 2kx8 dual port SRAM

- Two independent, asynchronous read/write ports
- Can access two different memory locations simultaneously
- Busy signal indicates if both ports attempt to access same location at the same time



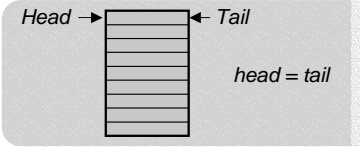


Building a FIFO

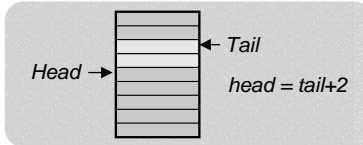


- Head pointer: points to next available location for writing
- Tail pointer: points to next available location for reading
- If head and tail pointers are the same, FIFO is empty
- If $head+1 = tail$, FIFO is full
- Pointers wrap around when they reach the end of the memory

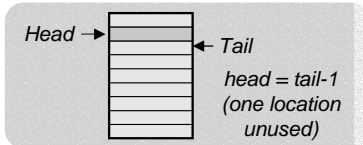
Empty



After 4 writes and 2 reads



Full



Building a FIFO

